Ecole Normale Supérieure de l'Enseignement Technique Mohammedia

Université Hassan II de Casablanca

ENSET

ا لمدرسة العليا لأساتذة التعليم التقني المحمدية

جامعة الحسن الثاني بالدار البيضاء

DÉPARTEMENT MATHÉMATIQUES ET INFORMATIQUE

Compte rendu

Développement mobile(android)

Filière :

« Génie du Logiciel et des Systèmes Informatiques Distribués »

GLSID

TP 1 : Codez votre propre calcultarice

ENSET Mohammedia

Réalisé par :

BOUJDI Mohamed

Année Universitaire : 2021-2022

Lien sur Github

Partie I : L'interface graphique de l'application:

L'organisation générale demandée peut se décomposer en un texte en haut de l'écran et un tableau de boutons en bas de l'écran. En utilisant le ConstraintLayout fourni par défaut,on peut donc placer un EditText en haut de l'écran sur l'intégralité de la largeur avec un texte aligné à droite et ainsi une gabarit TableLayout :

1. **Gabarit TableLayout:**

```
<TableLayout
    android:id="@+id/tableLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentBottom="true"
    android:layout_marginTop="32dp"
    android:shrinkColumns="*"
    android:stretchColumns="*"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/input"
    app:layout_constraintVertical_bias="0.96">
```

Chaque ligne de boutons du mon calculatrice doit être dans une balise TableRow:

```
<TableRow
    android:id="@+id/row1"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
```

2. **Composant EditText (screen):**

```
<EditText
    android:id="@+id/input"
    android:layout_width="413dp"
    android:layout_height="108dp"
    android:layout_marginTop="16dp"
    android:autofillHints="false"
    android:background="#38801A"
    android:inputType="none"
    android:minHeight="48dp"
    android:text="saisir votre opération"
    android:textAlignment="textEnd"
    android:textSize="36sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    tools:ignore="LabelFor" />
```

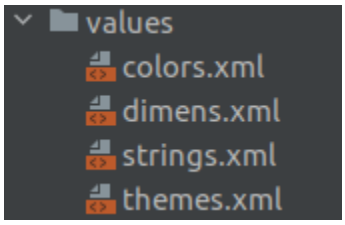
Le code xml qui définit les boutons de mon calculatrice (le bouton trois comme exemple):

```
<android.widget.Button
    android:id="@+id/btn3"
    style="@style/Widget.AppCompat.Button.Borderless"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_margin="3dp"
    android:layout_weight="1"
    android:onClick="_3btn"
    android:background="@drawable/circle"
    android:shadowColor="@color/black"
    android:text="3"
    android:textSize="20dp"
    android:outlineSpotShadowColor="@color/black"
    android:textColor="#FFFFFF" />
```

Ce bouton lié au logique java par cet méthode:

```
android:onClick="_3btn
```

Afin d'éviter les copiés/collés et de faciliter la maintenabilité du code, il est possible de définir une feuille de style définissant ce qu'est un bouton.



Partie II : Code JAVA:

La méthode **onCreate()** est utilisée pour démarrer l'activité.
super() est utilisé pour appeler le constructeur de la classe parent.
setContentView() est utilisé pour définir le xml.

```
EditText screen;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      screen =findViewById(R.id.input);
      screen.setShowSoftInputOnFocus(false);
      screen.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
        if(getString(R.string.saisir_votre_op_ration).equals(screen.getText())){
          screen.setText("");
        }
        }
      });

  }
```

La méthode **MAJscreen()** permet de mettre à jour le texte ;

```
private void MAJscreen(String strAajouter){
String oldString=screen.getText().toString();
int curseurPos=screen.getSelectionStart();
String leftStr=oldString.substring(0,curseurPos);
String rightStr=oldString.substring(curseurPos);
Boolean bool=getString(R.string.saisir_votre_op_ration).equals(screen.getText());
if(bool) {
  screen.setText(strAajouter);
  screen.setSelection(curseurPos + 1);
}else {
  screen.setText(String.format("%S%S%S", leftStr, strAajouter, rightStr));
  screen.setSelection(curseurPos + 1);
}
}
```

Les méthodes associées à des actions sur les boutons:

```
void zerobtn(View view) { MAJscreen( strAajouter: "0"); }
void _1btn(View view) { MAJscreen( strAajouter: "1"); }
void _2btn(View view) { MAJscreen( strAajouter: "2"); }
void _3btn(View view) { MAJscreen( strAajouter: "3"); }
void _4btn(View view) { MAJscreen( strAajouter: "4"); }
void _5btn(View view) { MAJscreen( strAajouter: "5"); }
void _6btn(View view) { MAJscreen( strAajouter: "6"); }
void _7btn(View view) { MAJscreen( strAajouter: "7"); }
void _8btn(View view) { MAJscreen( strAajouter: "8"); }
void _9btn(View view) { MAJscreen( strAajouter: "9"); }
void plusbtn(View view) { MAJscreen( strAajouter: "+"); }
void moinsbtn(View view) { MAJscreen( strAajouter: "-"); }
void multbtn(View view) { MAJscreen( strAajouter: "*"); }
void divbtn(View view) { MAJscreen( strAajouter: "/"); }
```

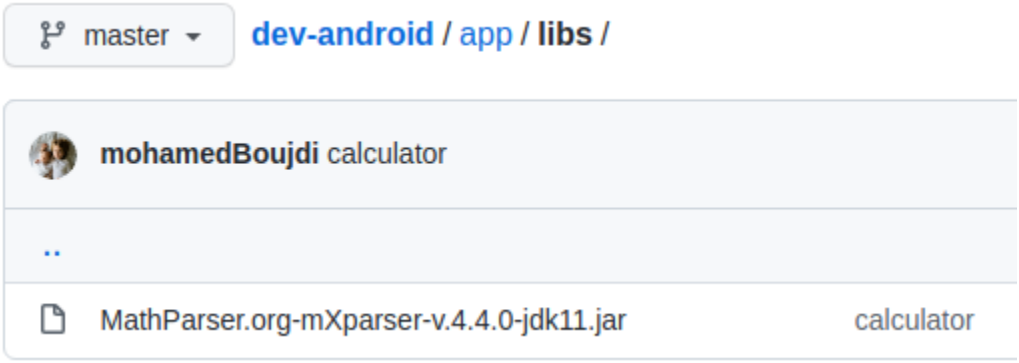
La méthode qui traite et ajoute les parenthèses:

```
void parenthesesbtn(View view){
  int curseurPos=screen.getSelectionStart();
  int openedParentheses=0;
  int closedParentheses=0;
  int textLength=screen.getText().length();
  for(int i=0;i<curseurPos;i++){
    if(screen.getText().toString().substring(i,i+1).equals("(")){
      openedParentheses++;
    }
    if(screen.getText().toString().substring(i,i+1).equals(")")){
      closedParentheses++;
    }
  }
  if(closedParentheses==openedParentheses
          ||screen.getText().toString().substring(textLength-1,textLength).equals("(")){
    MAJscreen( strAajouter: "(");
  } else if(closedParentheses < openedParentheses
          ||screen.getText().toString().substring(textLength-1,textLength).equals("(")){
    MAJscreen( strAajouter: ")");
  }
  screen.setSelection(curseurPos+1);

}
```

Libraire:

Je recourir à utiliser **mXparser** est un analyseur / évaluateur d'expressions mathématiques open source offrant la possibilité de calculer diverses expressions à la fois.



Les captures d'écran prise sur github

Dans mon code:

```
1   package com.example.moncalculatrice;
2
3   import androidx.appcompat.app.AppCompatActivity;
4
5   import android.os.Bundle;
6   import android.text.SpannableStringBuilder;
7   import android.view.View;
8   import android.widget.EditText;
9   import org.mariuszgromada.math.mxparser.*;
10
```

Les captures d'écran prise sur github

——----------

```
112    void egalbtn(View view){
113      String operation= screen.getText().toString();
114      operation.replaceAll("÷","/");
115      operation.replaceAll("×","*");
116      Expression expression=new Expression(operation);
117      String resultat=String.valueOf(expression.calculate());
118      screen.setText(resultat);
119      screen.setSelection(resultat.length());
120    }
```

Les captures d'écran prise sur github

Conclusion:

À partir de ce tp1 j'appris à réaliser **mon petit calculatrice** pour faire les opérations de base.